Einführung in Datenbanken

Übung 12: SQL-Wiederholung, Hinweise zur Klausur

Prof. Dr. Stefan Brass
PD Dr. Alexander Hinneburg
Martin-Luther-Universität Halle-Wittenberg

Wintersemester 2024/25

http://www.informatik.uni-halle.de/~brass/db24/

Inhalt

Organisatorisches (1)

- Haben Sie Fragen?

  Es ist ein wichtiger Zweck der Übung, Fragen zu beantworten.

- Diese Folien stehen morgen oder übermorgen auf der Webseite (Menupunkt „Übung“):

  [https://users.informatik.uni-halle.de/~brass/db24/uebung.html]

- Es wird noch eine Bonus-Aufgaben geben, bei der Sie ein SQL-Lernspiel ausprobieren können, das als Bachelorarbeit in meiner Gruppe entwickelt wurde.

  Es handelt sich um ein Text-Adventurespiel, bei dem die Daten in Tabellen stehen. Eine Web-Schnittstelle ist auch in Arbeit, aber aktuell einsetzbar ist ein Java-Programm mit GUI, das Sie auf Ihrem Rechner installieren müssten.

Inhalt

Präsenzaufgabe: Aggregationsanfrage (1)

- Schreiben Sie eine SQL-Anfrage zur Beantwortung der folgenden Frage.
- Von welchen Komponisten stehen mindestens 10 Musikstücke in der Datenbank?

name	vorname	count
Bach	Johann Sebastian	10
Händel	Georg Friedrich	25
Monteverdi	Claudio	10
Prokofiev	Serge	13
Telemann	Georg Philipp	15

- KOMPONIST(KNR, NAME, VORNAME°, GEBOREN°, GESTORBEN°)
- STUECK(SNR, KNR°→KOMPONIST, TITEL, TONART°, OPUS°)

Präsenzaufgabe: Aggregationsanfrage (2)

Lösung:

- Dies ist am besten mit `GROUP BY` und `HAVING` zu lösen:

```
SELECT k.name, k.vorname, COUNT(*) AS "count"
FROM   Komponist k, Stueck s
WHERE  k.knr = s.knr
GROUP  BY k.knr, k.name, k.vorname
HAVING COUNT(*) >= 10
ORDER  BY name, vorname
```

  Das `ORDER BY` war nicht verlangt.

- Link zum Adminer (Schema „`komponist_public`“):

  https://dbs.informatik.uni-halle.de/edb?pgsql=db&username=student_gast&db=postgres&ns=komponist_public

Präsenzaufgabe: Aggregationsanfrage (3)

Lösung, Forts.:

- Es geht auch mit einer Unteranfrage unter `FROM`:

```
SELECT k.name, k.vorname,
       a.anz_stuecke AS "count"
FROM   Komponist k,
       (SELECT knr, COUNT(*) AS anz_stuecke
        FROM   stueck
        GROUP  BY knr) a
WHERE  k.knr = a.knr
AND    a.anz_stuecke >= 10
ORDER  BY name, vorname
```

  Man braucht so kein `HAVING`. Aber man sollte `HAVING` kennen.

- 9 Zeilen statt 6 Zeilen bei der Standard-Lösung.

Präsenzaufgabe: Aggregationsanfrage (4)

Lösung, Forts.:

- Mit expliziter Hilfstabelle:

```
WITH anzahl(knr, anz_stuecke) AS
        (SELECT knr, COUNT(*) AS anz_stuecke
         FROM   stueck
         GROUP  BY knr)
SELECT k.name, k.vorname,
       a.anz_stuecke AS "count"
FROM   Komponist k, anzahl a
WHERE  k.knr = a.knr
AND    a.anz_stuecke >= 10
ORDER  BY name, vorname
```

Es wurde nur die Unteranfrage unter `WITH` verschoben. `WITH` ist normalerweise nur sinnvoll, wenn die Unteranfrage mehrfach verwendet wird (also hier nicht).

Inhalt

Hinweise zur Klausur (1)

- Alte Klausuren stehen auf der Webseite (unter „Prüfung“):

  [https://users.informatik.uni-halle.de/~brass/db24/pruefung.html]

- Die Aufgaben von Übungsblatt 11 stammen zum großen Teil aus der Zwischenklausur zu „Datenbanken I“ im WS 2003/2004.

  Die ersten drei Aufgaben sind daher eher unterhalb des Niveaus, das bei unserer Klausur vorausgesetzt wird (bei uns gibt es nur eine Klausur am Ende).

- In der Original-Klausur [.../db24/exams/exam7.pdf] gibt es noch eine Reihe interessanter Ankreuzaufgaben.

  Das Übungsblatt enthält dagegen nur SQL-Anfragen.

Hinweise zur Klausur (2)

- Bedenken Sie, dass Sie sich in der Klausur immer in ein neues DB-Schema einarbeiten müssen.
- Es empfiehlt sich, das Übungsblatt am Stück zu bearbeiten, und auch die Gesamtzeit zu messen.
- In der Klausur sind 10 min für jede SQL-Anfrage eingeplant (mit etwas Luft am Schluss, aber nicht sehr viel).

  Für das Übungsblatt mit 6 Anfragen sollten sie also möglichst nicht mehr als 60 min brauchen, 70 min wären auch möglich, 90 min wären schon zu viel.
- In der Klausur gilt die Regel „0 Punkte bei Syntaxfehlern“ nicht, aber in der Hausaufgabe.

  Die meisten Klausur-Abgaben mit Syntaxfehlern enthalten nicht nur einen einzigen Syntaxfehler, sondern mehrere Fehler, so dass es am Ende oft doch sehr wenig Punkte gibt.

Hinweise zur Klausur (3)

- Sie könnten also überlegen, wann es günstiger wäre, zur nächsten Aufgaben überzugehen, und das beim Probedurchlauf auch tun.

  Anschließend sollten Sie aber die Syntaxfehler noch entfernen, bevor Sie die Anfragen abgeben.

- Zeitmanagement ist auch eine Kompetenz, die bei der Klausur getestet wird.

  Sie sollten sich nicht in einer einzelnen Aufgabe verrennen, sondern spätestens nach 15 Minuten zur nächsten Aufgabe übergehen.

- ILIAS zeigt die verbleibenen Minuten an, und Sie können sich z.B. auf einem Schmierzettel notieren, wann Sie mit einer Aufgabe begonnen haben.

Hinweise zur Klausur (4)

- Nehmen Sie sich ein Blatt Schmierpapier und mehrere Stifte mit.

  Schmierpapier müssen Sie am Ende abgeben/wegschmeißen. Wir wollen nicht, dass Aufzeichnungen über die Klausur den Raum verlassen.

- Smartwatches sind bei der Klausur verboten (alles, was mehr kann, als Datum und Uhrzeit anzuzeigen).

- Alle elektronischen Geräte, die möglicherweise zur Kommunikation dienen könnten, sind verboten.

  Sie brauchen keinen Taschenrechner und dürfen auch keinen verwenden.

- Sie dürfen nichts im Ohr haben.

  Wenn Sie Ohropax oder ähnliches verwenden wollen, zeigen Sie das kurz zur Kontrolle vor. Wenn Sie eine Mütze oder Kopftuch tragen, kann verlangt werden, dass Sie das zur Kontrolle kurz abnehmen.

Hinweise zur Klausur (5)

- Nehmen Sie vor Beginn der Klausur alles, was Sie brauchen, aus Ihrem Rucksack/Ihrer Tasche, und stellen Sie es übersichtlich auf den Tisch.

  Wenn Sie nach Beginn der Klausur noch an Ihre Tasche müssen, rufen Sie eine Aufsichtsperson zur Kontrolle. Alle Gegenstände auf Ihrem Tisch müssen sichtbar sein (z.B. keine Kleidungsstücke, die andere Dinge darunter verdecken).

- Ihr Handy schalten Sie aus oder zumindest stumm und lassen es in Ihrem Rucksack/Ihrer Tasche.

  Wenn Sie unbedingt erreichbar sein müssen, melden Sie sich vor Beginn der Klausur bei der Aufsicht. Sie bekommen dann einen Platz ganz vorn. Beachten Sie, dass die Regeln für Handys bei verschiedenen Professoren unterschiedlich sind.

Hinweise zur Klausur (6)

- Bei der ersten Klausur (Montag, 03.03.2025, 10^{15}–12^{15}) sind 10 Seiten mit Notizen erlaubt.

  Im Format DIN A4. Entweder 5 Blätter doppelseitig bedruckt/beschrieben oder 10 Blätter einseitig.

- Beim zweiten Klausurtermin (Montag, 14.07.2025, 10^{15}–12^{15}) nur 6 Seiten.

  Weil mehr Vorbereitungszeit.

- Haben Sie schon angefangen, sich Notizen zu machen?
- Oder haben Sie eine passende „SQL Quick Reference“ im Internet gefunden, und die genau angeschaut?

Hinweise zur Klausur (7)

- Sie können die Anfragen in der Klausur mit dem Adminer testen.

  Wie aus den Hausaufgaben bekannt.

- Es zählt aber nur, was Sie in ILIAS abgeben.

  Sie brauchen also „Copy&Paste“. Leider hat ILIAS nur ein „Richt Text“ Eingabefeld, keins für Programmcode. Wir filtern unsichtbare Formatierung aus der Abgabe.

- Es empfiehlt sich, nur einmal auf dem Adminer-Link zu klicken, und damit einen neuen Tab (Registerkarte) zu öffnen.

  - Später können Sie dann mit `STRG+Tab` zwischen den Tabs wechseln.
  - Wenn Sie oft auf den Link klicken, öffnen Sie viele Tabs.

Hinweise zur Klausur (8)

- Weitere wichtige Keyboard-Shortcuts (stehen auch in der Klausur):
  - `STRG+Tab`: Tab/Registerkarte wechseln (ILIAS/Adminer)
  - `STRG+C`: Kopieren (in Zwischenablage)
  - `STRG+V`: Einfügen
  - `SHIFT+STRG+V`: Unformatiert einfügen
  - `STRG+Z`: Undo
  - `SHIFT+STRG+Z` oder `STRG+Y`: Redo
  - `SHIFT+Enter`: Normale neue Zeile in ILIAS (kein Paragraph).
- `STRG+A` für „alles markieren“ funktioniert nicht.

  Möglicher Datenverlust: Markierter Text wird durch nächstens Zeichen ersetzt.

Hinweise zur Klausur (9)

- Sie können alle Antworten noch ändern bis die 120 Minuten abgelaufen sind.

  Natürlich können Sie auch beliebig zwischen den Aufgaben springen.

- Es zählt die letzte gespeicherte Antwort.
- Alle Antworten werden auf dem Server gespeichert.
- Wenn Ihr Rechner ein Problem haben sollte, das die Aufsicht nicht beseitigen kann, können Sie an einen anderen Rechner wechseln und die Klausur dort fortsetzen.
- Datenschutz-Hinweis: Die Log-Datei des Adminers kann zur Plagiatskontrolle und anonymisiert für Forschungszwecke verwendet werden.

  Ebenso Log-Information von ILIAS.

Hinweise zur Klausur (10)

- Fragen Sie, wenn etwas nicht klar ist.
- Sie können auch probieren, bei völlig unverständlichen Fehlermeldungen von PostgreSQL (Adminer) zu fragen.

  Eventuell bekommen Sie aber keine Antwort (wenn wir denken, dass Sie es wissen müssen).
- Meist bekommt der Erste, der uns auf einen Fehler hinweist, einen Bonuspunkt (ohne Garantie).

  Das gilt auch für Lücken in der Aufgabenstellung (etwas, was nicht spezifiziert ist, aber wesentlich wäre). Wenn die alternative Interpretation der Aufgabenstellung uns aber sehr merkwürdig erscheint, gibt es keinen Bonuspunkt. Aber wir erklären Ihnen natürlich, wie es gemeint ist.

Hinweise zur Klausur (11)

- Zu jeder Anfrage wird das erwartete Ergebnis gezeigt.

  Wie bei den Hausaufgaben. Wenn keine spezielle Sortierung verlangt ist, ist das Beispiel-Ergebnis meist nach den ersten Spalten sortiert.
  Es gibt keinen Abzug für eine unverlangte (aber sinnvolle) Sortierung.

- Es bringt nichts, SQL-Anfragen zu basteln, die ausschließlich im Beispielzustand das richtige Ergebnis liefern, aber mit der Aufgabenstellung überhaupt nichts zu tun haben.

  Extreme Fälle werden mit 0 Punkten bewertet. Alle Anfragen werden auch manuell angeschaut, nicht nur automatisch korrigiert. Anfragen sind nur korrekt, wenn Sie die gewünschte Funktion von DB-Zuständen auf Ausgaben beschreiben.

- Auch minimale Syntaxfehler kosten wenigstens einen Punkt.

  Wenn Sie den Fehler schnell beseitigen können, beseitigen Sie ihn.
  Wenn Sie ihn nicht finden, die Aufsicht schon einmal gefragt haben, und die Zeit knapp wird, gehen Sie besser zur nächsten Aufgabe.

Hinweise zur Klausur (12)

- Anfragen sollen keine Duplikate liefern,
  - nicht nur im Beispielzustand,
  - sondern in jedem Zustand, der die Integritätsbedingungen (insbesondere Schlüssel) erfüllt.
- Ein fehlendes `DISTINCT` kostet einen halben Punkt.
- Ein überflüssiges `DISTINCT` kostet auch einen halben Punkt.

  Ebenso wie andere unnötige Komplikationen, z.B. ein unnötiger Join.
- Meist gibt es in Aufgaben, bei denen man über `DISTINCT` ernsthaft nachdenken sollte, einen Hinweis.

  Das bedeutet aber nicht automatisch, dass `DISTINCT` in diesem Fall richtig ist! Es könnte eine Falle sein. Selbst wenn es keine absichtliche Falle ist, könnte es sein, dass es mehrere mögliche Lösungen gibt, und der Dozent an eine andere Lösung gedacht hat als Ihre.

Hinweise zur Klausur (13)

- Nur Rechner, die von den Administratoren angeschaltet wurden, dürfen Sie verwenden.

  Z.B. soll immer ein Platz frei bleiben zwischen zwei Klausurteilnehmern. Eventuell werden bei kleineren Klausuren aber auch die hinteren Reihen frei gelassen. Schalten Sie keine Rechner selbst an, das gibt später nur Probleme.

- Setzen Sie sich möglichst nicht neben Ihre engsten Bekannten, mit denen Sie vielleicht in Versuchung kommen können, einen Täuschungsversuch zu unternehmen.

  Wir notieren uns, wer wo gesessen hat.

Hinweise zur Klausur (14)

- Das Übungsblatt enthält 6 SQL-Aufgaben.
- In der Klausur gibt es nur 5 SQL-Anfragen, dafür aber noch weitere Arten von Aufgaben, z.B.
  - Ankreuzaufgaben zur logischen Analyse von SQL-Anfragen,
  - Anfrage in der relationalen Algebra (eher einfach),
  - ER-Diagramm (DB-Entwurf),
  - Übersetzung vom ER-Modell ins relationale Modell,
  - Bonusaufgabe zu funktionalen Abhängigkeiten und BCNF.
- Bei Ankreuzaufgaben sollten Sie notfalls raten.
  Es gibt keine negativen Punkte.

Inhalt

Endklausur WS 2002/2003 (1)

- Steht als Beispiel 6 auf der Webseite der Vorlesung:

  [https://users.informatik.uni-halle.de/~brass/db24/exams/exam6.pdf]

- Datenbank für Bahnverbindungen aus meiner Zeit in Clausthal.

  Ich war dort nur ein Semester und habe noch bei Gießen gewohnt.

  Meist habe ich bei meinen Eltern in Braunschweig Station gemacht.

- Adminer:

  [https://dbs.informatik.uni-halle.de/edb-probeklausur-ws1920?pgsql=db&username=student_gast&db=postgres&ns=exam6]

Endklausur WS 2002/2003 (2)

- Die erste Tabelle enthält Daten über Züge. Züge fahren täglich, eventuell nicht samstags und sonntags.

  Kompliziertere Ausnahmen seien zur Vereinfachung nicht betrachtet.

FAHRPLAN		
ZNR	SA	SO
RB 25786	N	N
RB 25742	Y	Y
RB 34348	Y	N
RB 34303	Y	N
RB 34350	Y	N
RB 34305	Y	N

Endklausur WS 2002/2003 (3)

FAHRPLAN			
ZNR	BAHNHOF	ANKUNFT°	ABFAHRT°
RB 25786	Braunschweig	(null)	6:08
RB 25786	Wolfenbüttel	6:17	6:18
RB 25786	Goslar	7:18	(null)
RB 25742	Braunschweig	(null)	7:13
RB 25742	Wolfenbüttel	7:22	7:23
RB 25742	Goslar	8:01	(null)
RB 34348	Braunschweig	(null)	8:04
RB 34348	Salzgitter-Ringelheim	8:24	8:29
RB 34348	Holzminden	9:35	(null)
RB 34303	Salzgitter-Ringelheim	8:40	8:41
RB 34303	Goslar	8:54	(null)
⋮	⋮	⋮	⋮

Endklausur WS 2002/2003 (4)

- Die zweite Tabelle enthält Daten über Haltepunkte, d.h. wann welcher Zug in welchem Ort ankommt und abfährt.
- Beim Startbahnhof ist die Ankunft null, beim Zielbahnhof die Abfahrtzeit null.
- Die Spalten `ANKUNFT` und `ABFAHRT` seien vom Datentyp `TIME`, auf dem die üblichen Vergleiche definiert sind (z.B. `<`, `>`).
- Zur Vereinfachung nehmen wir an, dass kein Zug über Mitternacht hinaus fährt und wir auch nicht an entsprechenden Umsteige-Verbindungen interessiert sind.
- Nur die Spalten `ANKUNFT` und `ABFAHRT` erlauben Nullwerte.
- `ZNR` ist ein Fremdschlüssel, der auf `ZUEGE` verweist.

Endklausur WS 2002/2003 (5)

- Schema (hier nur zwei Tabellen, oft aber 3–4):
  - `ZUEGE(ZNR, SA, SO)`
  - `FAHRPLAN(ZNR, BAHNHOF, ANKUNFT°, ABFAHRT°)`
- Aufgabe:
  - Geben Sie alle direkten Verbindungen (d.h. ohne Umsteigen) von Braunschweig nach Goslar aus, die am Samstag fahren.

    D.h. die Spalte `SA` enthält ein `Y`.
  - Drucken Sie für jede solche Verbindung die `ZNR`, die Abfahrtszeit in Braunschweig und die Ankunftszeit in Goslar.

    Verbindungen von Goslar nach Braunschweig (d.h. in der umgekehrten Richtung) schließen Sie bitte aus (die Ankunftszeit in Goslar muss nach der Abfahrtszeit in Braunschweig sein).
  - Sortieren Sie die Ausgabe nach der Abfahrtszeit.

Endklausur WS 2002/2003 (6)

- Erwartetes Ergebnis (aus alter Klausur):

ZNR	ABFAHRT	ANKUNFT
RB 25742	7:13	8:01

Damals konnten die Anfragen nicht mit dem Adminer getestet werden.
Das Ergebnis wurde vermutlich manuell erstellt.

- Erwartetes Ergebnis (PostgreSQL im Adminer):

znr	abfahrt	ankunft
RB 25742	07:13:00	08:01:00

Falls tatsächlich eine Formatierung der Uhrzeit-Werte gewünscht gewesen wäre, hätte das in der Aufgabenstellung angesprochen werden müssen.
Außerdem wäre die entsprechende Formatierungs-Funktion genannt worden.
Gerade bei Datums- und Zeit-Werten wird nicht erwartet, dass Sie das auswendig wissen. Bei Zahlen möglicherweise schon (kam aber noch nie vor).

Endklausur WS 2002/2003 (7)

Lösung:

- Diese Anfrage erfordert einen Selbstverbund:

```
SELECT Z.ZNR, AB.ABFAHRT, AN.ANKUNFT
FROM   ZUEGE Z, FAHRPLAN AB, FAHRPLAN AN
WHERE  Z.ZNR = AB.ZNR
AND    Z.ZNR = AN.ZNR
AND    AB.BAHNHOF = 'Braunschweig'
AND    AN.BAHNHOF = 'Goslar'
AND    AB.ABFAHRT < AN.ANKUNFT
AND    Z.SA = 'Y'
ORDER  BY AB.ABFAHRT
```

Endklausur WS 2002/2003 (8)

- Dies wäre der Schwierigkeitsgrad, den Sie in der Klausur erwarten müssen (auch die übliche Länge).
- Alle Anfragen in der Klausur haben diesen Schwierigkeitsgrad.

  Es geht nicht etwa mit ganz simplen Anfragen los.
- Einige SQL-Konstrukte, die in der Klausur gebraucht werden:
  - `AND`, Vergleiche (`=`, `<>`, `>`, `>=`, `<`, `<=`), `DISTINCT`, `ORDER BY`
  - Rechenoperationen (`+`, `-`, `*`, `/`), Prozentrechnung, `ROUND`, `||`
  - Selbstverbund (wie hier)
  - `NOT EXISTS`, Unteranfragen, `>= ALL`
  - Aggregationsfunktionen, `GROUP BY`, `HAVING`
  - `LIKE`, `IS NULL`, `OR`
  - `UNION`, `LEFT JOIN`, `CASE`, `COALESCE`

Endklausur WS 2002/2003 (9)

Hinweis:

- Die Beispielklausur enthält noch mehr interessante Aufgaben.
- Es ist empfohlen, die anderen Aufgaben auch zu lösen.
- Üben Sie solange, bis Sie mit Aufgaben dieser Art keine Schwierigkeiten mehr haben!
- Der größte Teil der Klausuraufgaben kann keine Überraschung sein.

  Es ist natürlich jedes Mal ein anderes Schema und die konkreten Aufgaben sind neu, aber die Aufgabentypen sind sehr wahrscheinlich wie gehabt. Trotzdem waren die bisherigen Klausurergebnisse nicht wirklich befriedigend.

Inhalt

Aufgabe aus dem letzten Jahr: Fehlermeldungen

- Verwenden Sie das Schema „empdept_public“ im Adminer:
  - dept(deptno, dname, loc)
  - emp(empno, ename, job, mgr°→emp, hiredate, sal, comm°, deptno°→dept)
- Geben Sie drei Beispiele für fehlerhafte Anfragen und die zugehörige Fehlermeldung, die PostgreSQL ausgibt (3 Punkte).
  - Die Fehlermeldungen müssen unterschiedlich sein.

    Maximal ein „syntax error at or near“.
  - Die Fehler sollen gefühlt häufig vorkommen.

    Wenn die Fehlermeldung nicht klar ist, müssen Sie den Fehler erläutern.

    Bei der Klausur können Sie wertvolle Zeit sparen, wenn Sie schon mit Fehlermeldungen vertraut sind.
  - Die Beispiel-Anfragen sollten kurz sein.

    Sie sollen nur den Fehler demonstrieren.

Fehlermeldungen (1)

- Wenn man die Hausaufgaben selbst bearbeitet (und nicht abschreibt), sollte man bis zur Klausur eine ganze Anzahl Fehlermeldungen gesehen haben.

  Fehler passieren. Wichtig ist, sie vollständig aufzuklären und daraus zu lernen.

- Das ist nicht ein negativer Unglücksfall, sondern es gehört zum praktischen Umgang mit einem DBMS dazu, dass man
  - häufigere Fehlermeldungen schon mal gesehen hat,
  - versteht, was das Problem ist, und

    Z.B., weil man sich erinnert, was früher das Problem war.

  - den Fehler zügig korrigieren kann.
- Man kann auch mal bewusst falsche Anfragen eingeben.

Fehlermeldungen (1)

- Z.B. Syntaxfehler:

```
SELECT deptno, FROM dept
ERROR: syntax error at or near "from"
LINE 1: select deptno, from dept
                       ^
```

Die Markierung der Position bekommt man in `psql` angezeigt, aber leider nicht im Adminer. Stand der Technik in der Syntaxanalyse ist, dass der Fehler an der ersten Stelle gemeldet wird, an der keine gültige Fortsetzung mehr möglich ist. Natürlich kann der tatsächliche Fehler davor liegen.
Stand der Technik ist eigentlich auch, dass angegeben wird, was an dieser Stelle erwartet wird (was dort legal wäre). Diese Information gibt PostgreSQL leider nicht aus.

- Den Präfix „`Fehler in der SQL-Abfrage (7):`“ gibt der Adminer immer aus. Kann man wohl ignorieren.

Fehlermeldungen (2)

- Z.B. Syntaxfehler:

```
SELECT deptno FROM

ERROR: syntax error at end of input
LINE 2:
```

- Fehler im Tabellen-Namen:

```
SELECT deptno FROM deptx

ERROR: relation "deptx" does not exist
LINE 1: select deptno from deptx
                           ^
```

Fehlermeldungen (3)

- Fehler im Spalten-Namen:

```
SELECT deptnr FROM dept

ERROR: column "deptnr" does not exist
LINE 1: select deptnr from dept
                ^
HINT: Perhaps you meant to reference the column
      "dept.deptno".
```

Bei meiner alten PostgresSQL-Version (9.2.24) gibt `psql` den Tipp nicht mit aus (auch dann nicht, wenn ich den Konfigurationsparameter `client_min_messages` auf „`info`“ setze).

Fehlermeldungen (4)

- Spaltenreferenz nicht eindeutig:

```
SELECT deptno FROM emp, dept
```

```
ERROR: column reference "deptno" is ambiguous
LINE 1: select deptno from emp, dept
                ^
```

- Tupelvariable nicht deklariert:

```
SELECT e.ename FROM emp
```

```
ERROR: missing FROM-clause entry for table "e"
LINE 1: select e.ename from emp
                ^
```

Beide Fehler auf dieser Folie kamen in abgegebenen Lösungen in einer Klausur vor (in der man die Anfragen mit dem Adminer testen konnte). Dabei scheinen die Fehlermeldungen doch sehr klar.

Fehlermeldungen (5)

- Typfehler:

```
SELECT ename / 2 FROM emp
```

```
ERROR: operator does not exist: text / integer
LINE 1: select ename / 2 from emp
                     ^
HINT: No operator matches the given name
      and argument types.
      You might need to add explicit type casts.
```

PostgreSQL erlaubt überladene Funktionen und Operatoren. Deswegen die Aussage „kein Operator passt auf diesen Namen und Argumenttypen“, und nicht einfach: „Das erste Argument muss eine Zahl sein“.
Bei der Version dieser Datenbank im Adminer ist `ename` mit dem Typ `text` (fast unbegrenzte Zeichenketten) deklariert. Das ist nicht sehr portabel.
Weiteres Beispiel: `select round(ename) from emp`.
Fehlermeldung: „`function round(text) does not exist`“.

Fehlermeldungen (6)

- `GROUP BY` fehlt:

```
SELECT deptno, count(*)
FROM   emp
```

```
ERROR: column "emp.deptno"
       must appear in the GROUP BY clause
       or be used in an aggregate function
LINE 1: select deptno, count(*)
                ^
```

In Aggregationsanfragen können unter `SELECT` außerhalb von Aggregationsfunktionen nur `GROUP BY`-Attribute genutzt werden.

- Korrekt wäre:

```
SELECT deptno, count(*)
FROM   emp
GROUP  BY deptno
```

Fehlermeldungen (7)

- Aggregationsfunktion unter `WHERE`:

```
SELECT deptno
FROM   emp
WHERE  count(*) = 3
GROUP  BY deptno
```

```
ERROR:  aggregate functions are not allowed
        in WHERE
LINE 3: where count(*) = 3
                ^
```

- Korrekt wäre:

```
SELECT deptno
FROM   emp
GROUP  BY deptno
HAVING count(*) = 3
```

Inhalt

EMP-DEPT-Datenbank

- `dept`: „Department“ (Abteilungen einer Firma)

  `dept(deptno, dname, loc)`

- `emp`: „Employee“ (Angestellte der Firma)

  `emp(empno, ename, job, mgr°→emp, hiredate, sal, comm°, deptno°→dept)`

- `salgrade`: „Salary Grade“ (Gehaltsstufen)

  `salgrade(grade, losal°, hisal°)`

- Link zum Adminer (Schema „`empdept_public`“):

  https://dbs.informatik.uni-halle.de/edb?pgsql=db&username=student_gast&db=postgres&ns=empdept_public

Aufgabe 10.1: Abteilungs-Daten inkl. Budget (1)

- Geben Sie für jede Abteilung aus:
  - Nummer und Name der Abteilung,
  - die Anzahl der Angestellten in der Abteilung,
  - die Anzahl Angestellter mit Provision (d.h. kein Nullwert in `comm`),
  - das Gesamt-Budget für die Personalkosten der Abteilung, d.h. die Summe aller Gehälter (`sal`) und Provisionen (`comm`).
- **Sortieren** Sie das Ergebnis absteigend nach dem Budget, bei gleichem Budhet nach der Abteilungsnummer.

Aufgabe 10.1: Abteilungs-Daten inkl. Budget (2)

- Das erwartete Ergebnis ist:

deptno	dname	Angestelle	prov.	Budget
30	SALES	6	4	11600
20	RESEARCH	4	0	10075
10	ACCOUNTING	3	0	8750

- Bitte wählen Sie die Ausgabespalten wie im Beispiel.

  Die Spalte „`prov.`“ heißt eigentlich „`provisionsberechtigt`“.

- Leere Abteilungen sollen nicht in der Ausgabe erscheinen.

  Das vereinfacht die Aufgabe eher.

Aufgabe 10.1: Abteilungs-Daten inkl. Budget (3)

Lösung:

- Die Anfrage wird am besten mit `GROUP BY` gelöst:

```
SELECT d.deptno, d.dname,
       COUNT(*) AS "Angestelle",
       COUNT(comm) AS "provisionsberechtigt",
       SUM(SAL+COALESCE(comm,0)) AS "Budget"
FROM   emp e, dept d
WHERE  e.deptno = d.deptno
GROUP  BY d.deptno, d.dname
ORDER  BY "Budget" desc, deptno
```

Wenn man einfach `sal+comm` berechnet, ist das Ergebnis `NULL` für Angestellte ohne Provision (mit einem Nullwert in `comm`). Man muss also den Nullwert durch die Zahl `0` ersetzen.

Aufgabe 10.2: Job-Statistik (1)

- Geben Sie für jeden Job, den mindestens drei Angestellte haben, Folgendes aus:
  - die Anzahl Angestellter mit diesem Job,
  - die Anzahl Abteilungen, in denen der Job vorkommt, sowie
  - das minimale, maximale und durchschnittliche Gehalt aus.
- Runden Sie bitte das durchschnittliche Gehalt auf eine ganze Zahl und **sortieren** Sie die Ausgabe alphabetisch nach dem Job.

Aufgabe 10.2: Job-Statistik (2)

- Das erwartete Ergebnis ist:

job	Ang.	Abt.	von	bis	Durchschnitt
CLERK	4	3	800	1300	1038
MANAGER	3	3	2450	2975	2758
SALESMAN	4	1	1250	1600	1400

- Bitte wählen Sie die Ausgabespalten wie im Beispiel.

  Die Spalte „`Ang.`“ heißt eigentlich „`Angestellte`“ und die Spalte „`Abt.`“ eigentlich „`Abteilungen`“.

Aufgabe 10.2: Job-Statistik (3)

Lösung:

- Diese Anfrage kann gut mit `GROUP BY` und `HAVING` gelöst werden:

```
SELECT job, COUNT(*) AS "Angestellte",
       COUNT(DISTINCT deptno) AS "Abteilungen",
       MIN(sal) AS "von", MAX(sal) AS "bis",
       ROUND(AVG(sal)) AS "Durchschnitt"
FROM   emp
GROUP  BY job
HAVING COUNT(*) >= 3
ORDER  BY job
```

Aufgabe 10.3: Gehaltsklassen pro Abteilung (1)

- Geben Sie für jede Abteilung Folgendes aus:
  - Nummer und Name der Abteilung,
  - das durchschnittliche Gehalt (gerundet auf eine ganze Zahl),
  - die Anzahl Angestellter mit Gehalt bis 2000 $,
  - die Anzahl Angestellter mit Gehalt über 2000 $.
- Das erwartete Ergebnis ist:

deptno	dname	AVG	bis 2000	ueber 2000
10	ACCOUNTING	2917	1	2
20	RESEARCH	2519	1	3
30	SALES	1567	5	1

Aufgabe 10.3: Gehaltsklassen pro Abteilung (2)

- Wählen Sie die Spaltenüberschriften bitte wie im Beispiel gezeigt.

  Die Spalte „`AVG`“ heißt eigentlich „`Durchschnitt`“.

- Eine spezielle Sortierung ist nicht verlangt.

  Wenn Sie sich den Vergleich mit der Beispiel-Ausgabe erleichtern wollen, sortieren Sie nach `deptno`.

Aufgabe 10.3: Gehaltsklassen pro Abteilung (3)

Lösung:

- Hier kann man einen Trick mit CASE verwenden, und die jeweils nicht zu zählenden Zeilen auf einen Nullwert abbilden:

```
SELECT d.deptno, d.dname,
       ROUND(AVG(sal)) AS "Durchschnitt",
       COUNT(CASE WHEN SAL <= 2000 THEN 1 END)
           AS "bis 2000",
       COUNT(CASE WHEN SAL > 2000 THEN 1 END)
           AS "ueber 2000"
FROM   dept d, emp e
WHERE  d.deptno = e.deptno
GROUP  BY d.deptno, d.dname
ORDER  BY d.deptno
```

Aufgabe 10.3: Gehaltsklassen pro Abteilung (4)

Lösung, Forts.:

- Wenn man im `CASE` kein `ELSE` angibt, ist der Default `ELSE NULL`.
- Bei der Lösung spielt es keine Rolle, welchen Nicht-Nullwert man wählt. Statt `1` hätte man auch `27` schreiben können oder `'zaehlen'`.
- Wenn man sich an den Trick mit `CASE` nicht erinnert, kann man auch Unteranfragen unter `SELECT` einsetzen, wie auf der nächsten Folie gezeigt.

  Das ist das deutlich länger (es passt nur mit Auslassungen auf die Folie). Es gibt auch Code-Duplizierung für jeden Wert, den man unter `SELECT` berechnen möchte. Mit der `EXISTS`-Bedingung muss man prüfen, dass die Abteilung nicht leer ist. Sonst bekommt man auch Abteilung `40 OPERATIONS`.

Aufgabe 10.3: Gehaltsklassen pro Abteilung (5)

Lösung, Forts.:

- Lösung mit Unteranfragen unter SELECT:

```
SELECT d.deptno, d.dname,
       (SELECT ROUND(AVG(sal))
        FROM   emp e
        WHERE  e.deptno = d.deptno)
           AS "Durchschnitt",
       (SELECT COUNT(*)
        FROM   emp e
        WHERE  e.deptno = d.deptno
        AND    SAL <= 2000) AS "bis 2000",
       ... AS "ueber 2000"
FROM   dept d
WHERE  EXISTS(SELECT * FROM emp e WHERE ...)
```

Aufgabe 10.4: Maximale Anzahl Untergebene (1)

- Geben Sie den oder die Angestellten mit der maximalen Anzahl (direkter) Untergebener aus.

  Die Angestellten-Nummer des direkten Vorgesetzten steht in der Spalte `mgr`.

- Geben Sie Nummer, Name und Job des Vorgesetzten, sowie die Anzahl direkter Untergebener aus.

- Das erwartete Ergebnis ist:

empno	ename	job	Untergebene
7698	BLAKE	MANAGER	5

Aufgabe 10.4: Maximale Anzahl Untergebene (2)

Lösung:

- Lösung mit `HAVING` und `>= ALL`:

```
SELECT v.empno, v.ename, v.job,
       COUNT(*) AS "Untergebene"
FROM   emp v, emp u
WHERE  v.empno = u.mgr
GROUP  BY v.empno, v.ename, v.job
HAVING COUNT(*) >= ALL (SELECT COUNT(*)
                        FROM   emp
                        GROUP  BY mgr)
```

Aufgabe 10.4: Maximale Anzahl Untergebene (3)

Lösung, Forts.:

- Alternative mit Hilfstabelle, die zu jedem Angestellten die Anzahl Untergebenen enthält:

```
WITH   anz_unt(empno, anz) AS
       (SELECT mgr, COUNT(*)
        FROM   emp
        GROUP  BY mgr)
SELECT v.empno, v.ename, v.job,
       a.anz AS "Untergebene"
FROM   emp v, anz_unt a
WHERE  v.empno = a.empno
AND    anz = (SELECT MAX(anz)
              FROM   anz_unt)
```

Aufgabe 10.5: Nummerierung der Angestellten (1)

- Nummerieren Sie die Angestellten durch (1, 2, ...), und zwar nach dem Gehalt (größtes zuerst), und bei gleichem Gehalt nach dem Namen.

  Sie können diese Aufgabe lösen, indem Sie für jeden Angstellten zählen, wie viele Angestellte (A) mehr verdienen als der aktuelle Angestellte, oder (B) genauso viel verdienen wie der aktuelle Angestellte, aber einen Namen haben, der alphabetisch vor dem des aktuellen Angestellten kommt (oder gleich ist). Bei dieser Aufgabe wird angenommen, dass die Angestellten-Namen eindeutig sind.

- Geben Sie die berechnete Nummer, den Namen und das Gehalt für jeden Angestellten aus.
- **Sortieren** Sie die Ausgabe nach der Nummer.
- Die erwartete Antwort steht auf der nächsten Folie.

Aufgabe 10.5: Nummerierung der Angestellten (2)

Nr	ename	sal
1	KING	5000
2	FORD	3000
3	SCOTT	3000
4	JONES	2975
5	BLAKE	2850
6	CLARK	2450
7	ALLEN	1600
8	TURNER	1500
9	MILLER	1300
10	MARTIN	1250
11	WARD	1250
12	ADAMS	1100
13	JAMES	950
14	SMITH	800

Aufgabe 10.5: Nummerierung der Angestellten (3)

- Lösung, die die Anzahl von Angestellten zählt, die in der Sortierreihenfolge vor dem aktuellen Angestellten kommen:

```
SELECT COUNT(*) AS "Nr", e.ename, e.sal
FROM   emp e, emp vorher
WHERE  e.sal < vorher.sal
OR     (e.sal = vorher.sal
        AND e.ename >= vorher.ename)
GROUP  BY e.empno, e.ename
ORDER  BY "Nr"
```

Die Bedingung für `vorher` wird auch vom aktuellen Angestellten `e` erfüllt, sonst würde der erste Angestellte in der Ausgabe fehlen. So bekommt er korrekt die Nummer 1, weil er selbst als `vorher` eingesetzt werden kann.